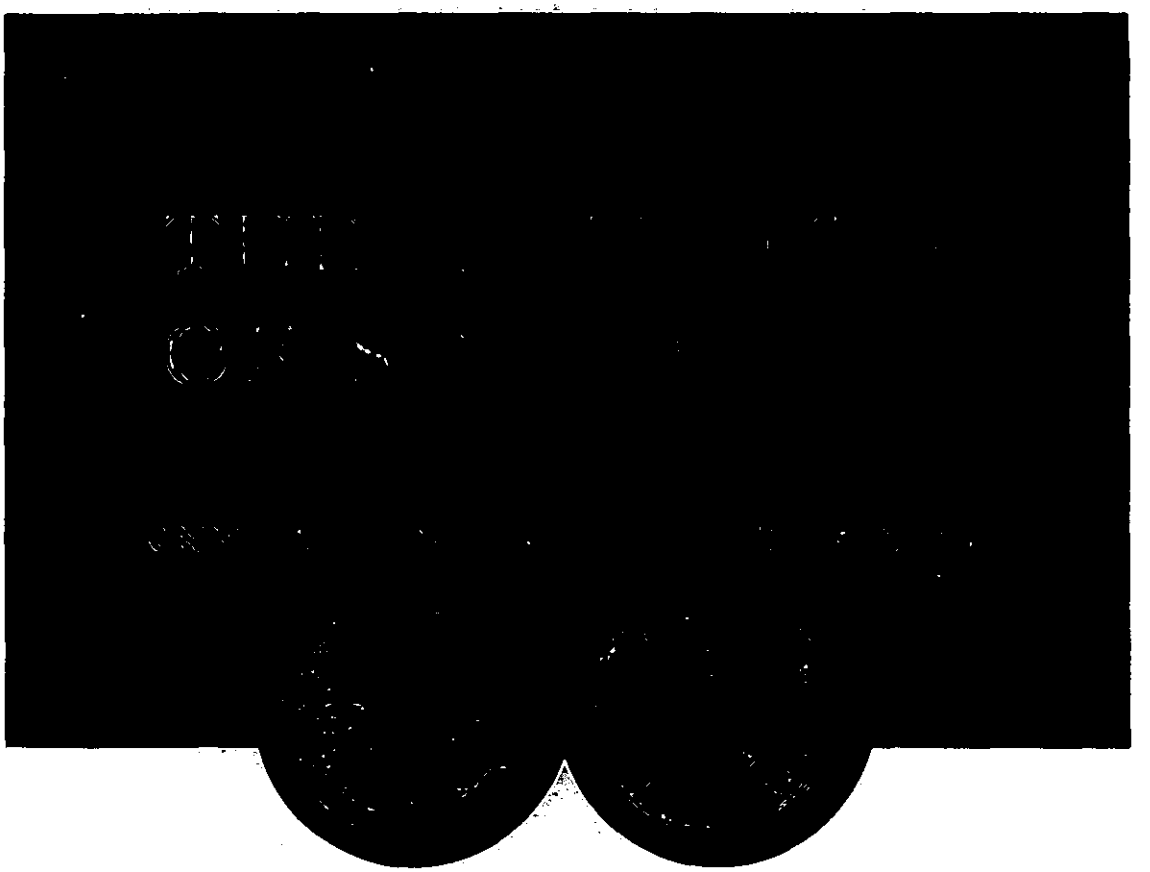


An Algorithm for the Construction of Spatial Coverage Designs with Implementation in SPLUS

by

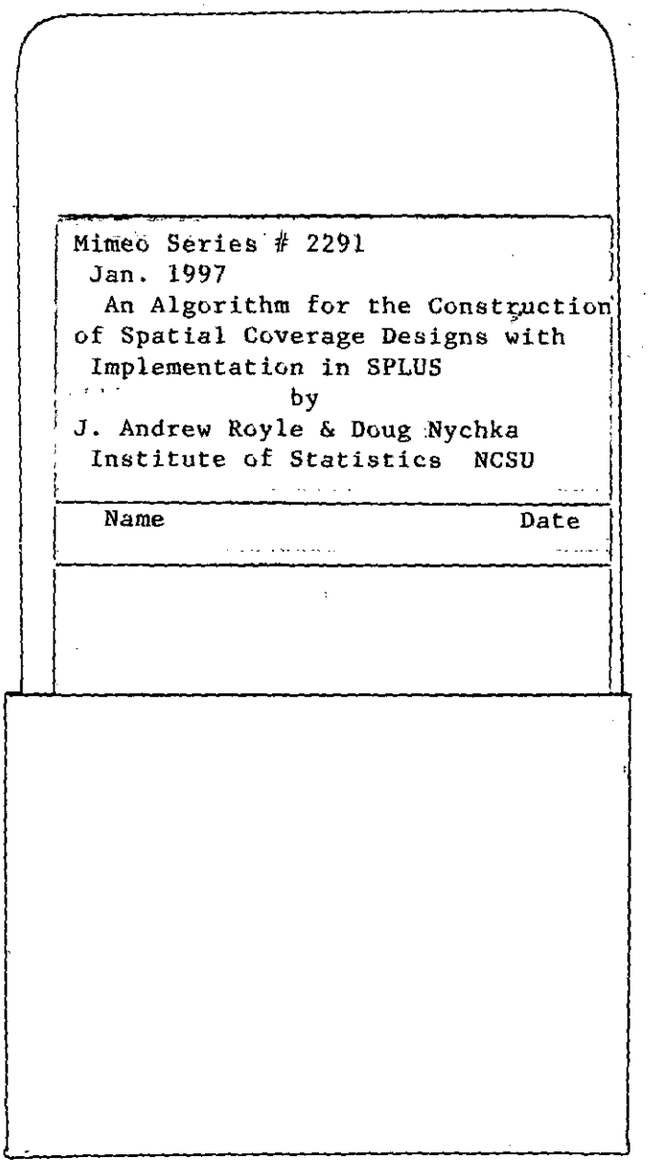
J. Andrew Royle
and
Doug Nychka

Institute of Statistics Mimeo Series Number 2291

January, 1997

NORTH CAROLINA STATE UNIVERSITY
Raleigh, North Carolina

Department of Statistics Library



Mimeo Series # 2291
Jan. 1997

An Algorithm for the Construction of Spatial Coverage Designs with Implementation in SPLUS

by
J. Andrew Royle & Doug Nychka
Institute of Statistics NCSU

Name Date

An Algorithm for the Construction of Spatial Coverage Designs with Implementation in SPLUS

J. Andrew Royle*
Geophysical Statistics Project
National Center for Atmospheric Research
Doug Nychka
Department of Statistics
North Carolina State University

Abstract

Space-filling "coverage" designs are spatial sampling plans which optimize a distance-based criterion. Because they do not depend on the covariance structure of the process to be sampled, coverage designs are more efficiently computed than designs which are optimal for mean squared error criteria. This paper presents an efficient algorithm for the construction of coverage designs and evaluates it's performance in terms of computation time and effectiveness at finding "good" designs. Results suggest that near-optimal designs for reasonably large problems can be computed very efficiently. The algorithm is implemented in the statistical programming language SPLUS, and examples of the construction of coverage designs are given involving an existing network of ozone monitoring sites.

Keywords: Spatial design, space-filling designs, spatial statistics, spatial sampling, network design.

*Address: Box 3000, Boulder, Colorado 80307. (303) 497-1704.

1 Introduction

A practical problem in spatial statistics is that of optimizing the location of sampling points, such as in the construction of air-quality monitoring networks.

The problem may be approached by specifying a covariance-based criterion and optimizing with respect to the sample locations. For example one could find the configuration of locations so that the average or maximum of the prediction variance over a region of interest is minimized. There are two basic problems with this approach. First, one must know the covariance function of the process for which the design is meant to sample, and misspecification produces designs which are not optimal. Often, the reason for sampling is because there is little prior knowledge of the field over a particular region. Thus specifying a covariance model in order to construct the design may be problematic. Second, minimization of these criteria based on prediction variance is difficult due to the computation of the criterion. For a fixed set of locations calculation of the prediction variance involves a Cholesky decomposition of a covariance matrix of size n. This decomposition must be recomputed as locations are varied. Moreover, it is difficult to find the gradient of these criteria as part of a steepest descent algorithm. See Cox et al. (1995) for a review of environmental sampling network design.

An alternative method of constructing spatial designs is to base the design on a geometric criterion that does not involve the covariance structure of the process. Nychka et al. (1997) considered "space-filling" designs, whereby sampling points are located so as to minimize a criterion that is only a function of the distance between the sampling locations and non-sampling locations. This criterion, henceforth called the "coverage" criterion, measures how well a set of sample locations (i.e. the "design") *covers* the domain of interest. Saltzman et al. (1996) have shown that coverage designs closely approximate those that minimize the average prediction variance, and Nychka et al. (1997) provided some empirical evidence that they perform comparable to designs which minimize the prediction variance of the average.

Some practical experience with thinning a large network for monitoring ozone can be found in Nychka et al. (1997). Thus there is some evidence that coverage designs compromise little with regard to design performance and promise to be very useful. The coverage criterion can be computed efficiently, and an efficient algorithm that is based on "point-swapping" (also called "exchange" algorithms) can be used to optimize over all possible designs. Although this algorithm is gradient free we have been able to improve its efficiency by restricting the swapping to nearest neighbors.

The purpose of this paper is to describe this algorithm for computing a coverage design and it's implementation in the software package SPLUS. SPLUS is an objected-oriented statistical programming language ideally suited for such applications, due to it's graphical capabilities and the ease at which output from a process may be manipulated in conjunction with other functions. In Section 2 we define coverage designs and the algorithm for their construction is described in Section 3. The algorithm is evaluated in terms of performance and run-time in Section 4. Examples of constructing coverage designs for an existing network of ozone monitoring sites in Chicago are given in Section 5. Conclusions and discussion are given in Section 6, and some SPLUS examples, and the SPLUS help file for the function `cover.design`, are given in Section 7.

2 Space filling designs

As an alternative to covariance-based optimal design, we consider a geometric criterion to locate design points. It will be assumed that the designs are the solution to the discrete problem of selecting a subset of points from a larger set of candidates. Let $\mathcal{C}$ denote the set of N candidate points and $\mathcal{D}$ a subset constituting a design of size n and let $p < 0$.

A metric for the distance between a point and a particular design is

$$d_p(\mathbf{x}, \mathcal{D}) = \left(\sum_{\mathbf{u} \in \mathcal{D}} ||\mathbf{x} - \mathbf{u}||^p \right)^{(1/p)}.$$

This criterion can be thought of as measuring how well the design *covers* the point $\mathbf{x}$. For $p < 0$ it is easy to show that $d_p(\mathbf{x}, \mathcal{D}) \to 0$ as $\mathbf{x}$ converges to a member of $\mathcal{D}$. Taking $q > 0$, an overall coverage criterion is an L_q average of "coverages" for each candidate point

$$C_{p,q}(\mathcal{D}) = \left(\sum_{u \in \mathcal{C}} d_p(\mathbf{x}, \mathcal{D})^q \right)^{(1/q)} . \tag{1}$$

The coverage design for a given size is the subset that minimizes $C_{p,q}(\mathcal{D})$ for all $\mathcal{D} \subset \mathcal{C}$. In the limit as $p \to -\infty$ and $q \to \infty$, $C_{p,q}$ converges to the criterion commonly associated with minimax space filling sets of points, and these designs are computed by the Gosset design package (Hardin and Sloane, 1994). (The terminology is confusing because $C_{-\infty,\infty}$ is actually maximum over minimums). See also Johnson et al. (1990) for some theoretical connections between space filling designs and those based on prediction error for a spatial process. In general, coverage designs can be computed by the SAS procedure PROC OPTEX (Tobias, 1995).

In this work, coverage designs were found with $p = -5$ and $q = 1$. Based on the examples of Tobias, this choice was a compromise between designs that were close to the minimax solution but, because p is finite, were more stable to compute.

3 A Point Swapping Algorithm

The algorithm for finding the optimum design based on the coverage criterion uses random starting configurations and decreases the coverage criterion by swapping a candidate point with a design point. The use of point-swapping or exchange algorithms is not new. Two early references are Kennard and Stone (1969) and Mitchell (1974). See also Marengo and Todeschini (1992) and Tobias (1995) for application to distance-based criteria.

The basic idea of the algorithm is simple. For a given point in the current design, replace this point with members of the candidate set. If a particular swap reduces the coverage criterion over the initial design then this new point is included in the

design and the old point is moved to the candidate set. This process is repeated for each member of the design set until there are not longer any productive swaps. Note that when the design is modified the coverage criterion will always be reduced and so the algorithm will always converge to some solution.

The most important detail in this algorithm is efficiently computing the new coverage criterion when two points are swapped. Following the definition of the coverage criterion (Equation 1) for fixed parameters p and q, let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ denote the N candidate points and $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$ the n design points. Let $\mathbf{D}$ be the $N \times n$ matrix such that $\mathbf{D}_{ij} = ||\mathbf{x}_i - \mathbf{y}_j||^p$. We also need the vector of row sums, $\mathbf{r}$, with

$$r_i = \sum_{j=1}^{n} D_{ij}$$

Then, the coverage criterion simplifies to

$$C_{p,q}(\mathcal{D}) = (\sum_{i=1}^{N} (r_i)^{q/p})^{1/q}.$$

Now consider the design, $\mathcal{D}^*$, where point $\mathbf{y}_{i'}$ is replaced by the candidate point $\mathbf{x}_{j'}$. To compute the corresponding criterion for $\mathcal{D}^*$, one need only update the i'th row and the j'th column of $\mathbf{D}$. Thus, out of $N \times n$ elements of the matrix $\mathbf{D}$, only $N + n - 1$ elements need to be recomputed. Moreover if $\mathbf{r}^*$ is the row sum vector for the new design then for $i \neq i'$

$$r_i^* = r_i - ||y_{i'} - x_i||^p + ||x_{j'} - x_i||^p$$

and for $i = i'$

$$r_i^* = \sum_{j \neq j'} ||y_j - y_{i'}||^p + ||x_{j'} - y_{i'}||^p.$$

Of course with the new row sums the coverage criterion reflecting this swap is

$$C_{p,q}(\mathcal{D}^*) = (\sum_{i=1}^{N} (r_i^*)^{q/p})^{1/q}.$$

Thus, for swapping a single candidate point for a single design point, computation involves finding $N + n$ elements of D to update the row sums and then summing the powers of the N terms in $C(p, q)$.

The algorithm may be summarized as follows:

(1) Select a starting design, compute $\mathbf{r}$ and $C_{p,q}(\mathcal{D})$.

(2) For each $\mathbf{x}_i \in \mathcal{D}$ that are not fixed,

(3) Replace $\mathbf{y}_j$ by $\mathbf{x}_i : i = 1, 2, \ldots, N$ and compute N criterion values.

(4) Swap $\mathbf{x}_i$ with the $\mathbf{y}_j$ which produces the largest decrease over the initial criterion. Given this swap recompute $\mathbf{r}$ and $C_{p,q}(\mathcal{D})$.

(5) Repeat 2-4 until no swap can be made.

This algorithm will always converge, but it may not produce the optimal design and so it is important to use several different starting designs. Because of the inner loop over design points it is simple to consider fixed design points in the optimization. These points are specified in the initial configuration and are not swapped out in step 2. One advantage of simple swapping is that no assumptions are made on the shape or spacings of points in the candidate set. This is useful in applications where the design region is irregular reflecting perhaps geographical boundaries or areas where it is not possible to take measurements. Finally it should be noted that this algorithm does not depend on a specific metric to measure coverage. For example, in geographic coordinates, great circle distance may be more appropriate to measure distances between points than Euclidean norm. Our implementation of this algorithm in SPLUS, in a function called `cover.design`, allows the user to supply an arbitrary distance function in the form of an S function. The function `cover.design` and it's options and output are described in Section 7.

An important short-cut that appears to decrease the run time substantially is use of a nearest-neighbor search, where, for each $\mathbf{y}_j$ only it's M nearest neighbors are considered for swapping. Issues of convergence to the optimal and run time are discussed in the following section.

4 Evaluation of the Algorithm

The point-swapping algorithm discussed in the preceeding section does not always converge to the optimal design. Therefore, in order to have some confidence in this procedure it is helpful to quantify the probability of achieving the optimal design, or at least a "good" design (i.e. one that is sufficiently close to the optimal). One can use this information to assist in selecting the number of random starting configurations over which to optimize. The performance of the algorithm depends on both the number of candidate points, and the design size. Here we study these factors for a 4×4 square region containing 25, 81, and 289 candidate points, and for various design sizes. We also examine the run-time required to compute the space filling designs, as well as the effect of employing a nearest-neighbor search strategy. All computations were done on a dedicated `Sparc Ultra` processor.

For each level of discretization ($N = 25, 81, 289$) and design size ($n = 4, 5, \ldots, 20$), optimization was carried out using 500 random starting configurations, resulting in (possibly) 500 different designs. For a given optimization, define the "best" design as that with the minimum criteria from the optimization procedure. The "optimal" design is some theoretical design which may or may not be the same as that achieved from the optimization. We are interested in (1) the best designs for the 500 optimizations and, in particular, how frequently a common best design is observed, (2) how close the 500 designs are to the overall best one, on average, and (3) how much improvement is observed in the criterion by using the point-swapping algorithm. In general, we will not observe *the* optimal design, however if the algorithm tends to converge to a common best design frequently, we can have some assurance that we've achieved the true optimal. To assess (2), we will define a criterion which measures the average closeness of the 500 designs to the overall best. Denote the 500 (optimized) criterion values as $c_i : i = 1, 2, \ldots, 500$, then this measure of average

closeness to the optimal design (ACO) is:

$$ACO = [\frac{1}{500}\sum_{i=1}^{500}\frac{c_i - \min_i c_i}{\min_i c_i}] \times 100.$$

This is just the average percent deviation from the best criterion value for the 500 optimized values. For (3), let $c_{oi} : i = 1, 2, \ldots, 500$ be the coverage criterion corresponding to the ith random starting design. Define the average percent improvement (API) in the coverage criterion as:

$$API = [\frac{1}{500}\sum_{i=1}^{500}\frac{c_{oi} - c_i}{c_{oi}}] \times 100.$$

Incorporating a nearest neighbor search strategy greatly decreases the time necessary to minimize the coverage criterion. Experiments have been conducted looking at various neighborhood sizes, and the results indicated that the smallest possible neighborhood size (e.g. 4 neighbors on a regular lattice) does not perform well, whereas "larger" (but much smaller than N) neighborhoods perform comparable to a search over all candidate points. Optimizations under the above scenarios were also done using a neighborhood size representing $\approx 25\%$ of the available candidate points (i.e. 8, 24 and 80).

Values of ACO and API are given in Table 1. Boxplots of the starting and optimal criteria for $N = 25, 81, 289$ are given in Figures 1-3. Also, the number of times that the best design was found is shown. The run-time to perform 10 optimizations for each situation using both the search over the full candidate set and the nearest-neighbor search is given in Table 2.

Results from Table 1 and Figures 1-3 indicate that for many cases (e.g. when N and/or n are large), the swapping algorithm does not find a common minimum very often. This raises the obvious question of whether the observed best design is *the* optimal design, or if it is even suitably "close". Nevertheless, the best designs tended to be very close to one another in terms of the criterion value, suggesting that they are near-optimal (i.e. little decrease in criterion could be achieved). The ACO

criterion indicates that the 500 best designs for each case tend to be within 1.5% of the minimum of all 500 in terms of the coverage criterion. Note that there is a large improvement in criterion value as a result of optimization, and this improvement increases substantially with N. For $N = 289$, we see generally a $> 30\%$ decrease in the criterion value, on average and for all p. Use of a nearest neighbor search strategy greatly reduces computation time while still finding designs that are nearly equivalent to the search over all candidate points. In a few instances, the nearest neighbor search found slightly better designs. For the largest problems involving the 289 candidate points, optimization takes an average of between 1 and 4 minutes, depending on design size (Table 2). This is reduced by $50-80\%$ (for larger values of N) when the nearest-neighbor search strategy is implemented. This suggests that one could effectively optimize much larger problems in a reasonable amount of time. Indeed, the example given in the following section is considerably larger than the situations just examined.

Since computation of the random starting design and it's corresponding coverage criterion value is computationally cheap, we looked at the possibility that "better" starting designs lead to better end designs. No significant relationship between the criterion value of the starting design and that of the best design was found.

5 Example: An Ozone Monitoring Network

Here we illustrate the construction of coverage designs in a region with an existing network of 21 ozone monitoring sites in the Chicago area. The discretized Chicago "region" is a grid of 720 points located in the convex hull of the existing network but excluding the area over Lake Michigan (see Figure 4(a)).

First, consider reducing the network size from 21 to 5. The best 5 site network based on the coverage criterion is shown in Figure 4(b). For comparison, the best 5 site network over the Chicago region is shown in Figure 4(c). Note that several of the best sites chosen from the full candidate set are not that close to any of the

Table 1: Summary of coverage designs based on 500 random starts using 3 candidate sets (25,81,289), and 2 search strategies ("all" = search over all available candidates and "nn" = search over approximately 10% of the nearest neighbors). ACO is the average (percent) deviation from the best observed design and API is the average (percent) improvement from the random starting design. "–" indicates that the optimization could not be carried out for that particular case.

	Number of candidates (N)											
	N=25				N=81				N=289			
	ACO		API		ACO		API		ACO		API	
n	all	nn	all	nn	all	nn	all	nn	all	nn	all	nn
5	0.6	1.0	22.3	23.1	0.3	0.5	32.4	32.3	0.9	1.0	35.2	34.4
6	0.8	1.1	19.2	19.0	0.7	0.8	31.4	32.4	0.6	0.8	35.0	34.1
7	1.7	1.9	17.0	16.9	0.6	0.9	32.3	31.3	0.5	0.7	33.4	33.2
8	1.0	1.3	15.6	15.6	1.5	2.0	32.2	31.7	1.0	1.1	34.6	34.4
9	1.3	1.6	14.7	14.6	2.0	3.2	33.4	32.5	1.0	1.4	35.1	34.4
10	0.9	1.0	13.6	13.6	1.7	2.1	31.3	29.3	0.9	1.1	34.1	32.9
11	1.2	1.3	13.2	12.6	1.4	1.8	29.0	27.8	1.0	1.1	32.1	31.9
12	2.2	2.3	12.8	12.7	1.3	1.9	27.6	26.1	0.9	1.2	31.5	32.6
13	2.5	2.7	12.7	11.9	1.2	1.7	25.0	24.3	0.7	1.0	31.0	31.2
14	1.5	1.7	11.7	11.7	1.2	1.7	24.6	23.1	1.0	1.2	31.4	31.2
15	0.9	1.0	10.9	10.5	1.2	1.7	23.1	22.3	1.1	1.5	31.1	30.3
16	0.9	1.0	9.9	9.7	1.2	1.6	21.3	21.8	1.2	1.5	31.5	29.8
17	0.6	0.5	9.3	9.1	1.3	1.5	21.6	20.6	1.0	1.3	29.9	29.5
18	0.6	–	8.2	–	1.6	1.6	19.9	20.3	1.0	1.4	29.6	29.5
19	0.8	–	7.9	–	1.3	1.7	18.9	18.8	1.0	1.2	29.1	29.7
20	1.2	–	7.7	–	1.2	1.6	18.4	18.2	1.1	1.3	29.7	29.5

Table 2: Time required to perform 10 optimizations using random starting configurations for each case and searches over all possible candidate points and a nearest-neighbor search using approximately 25% of the available candidate points. For nearest-neighbor search, % decrease in run time is also given. Times are "elapsed time" in seconds from the command `unix.time`.

	All candidate search			Nearest neighbor search					
n	25	81	289	25	% decrease	81	% decrease	289	% decrease
5	25	117	678	13	48.0	48	59.0	230	66.1
6	24	117	773	12	50.0	62	47.0	278	64.0
7	26	227	854	18	30.8	62	72.7	332	61.1
8	33	171	1298	14	57.6	65	62.0	357	72.5
9	32	254	1547	15	53.1	89	65.0	413	73.3
10	27	231	1777	20	25.9	68	70.6	442	75.1
11	33	235	1683	22	33.3	80	66.0	434	74.2
12	37	251	1437	21	43.2	96	61.8	599	58.3
13	29	252	1666	21	27.6	108	57.1	493	70.4
14	29	154	1744	25	13.8	93	39.6	505	71.0
15	28	246	2633	28	0.0	96	61.0	570	78.4
16	26	261	1830	23	11.5	117	55.2	580	68.3
17	25	237	2500	25	0.0	94	60.3	588	76.5
18	27	261	2173	–	–	98	62.5	769	64.6
19	21	193	1811	–	–	96	50.3	897	50.5
20	17	236	1923	–	–	124	47.5	765	60.2

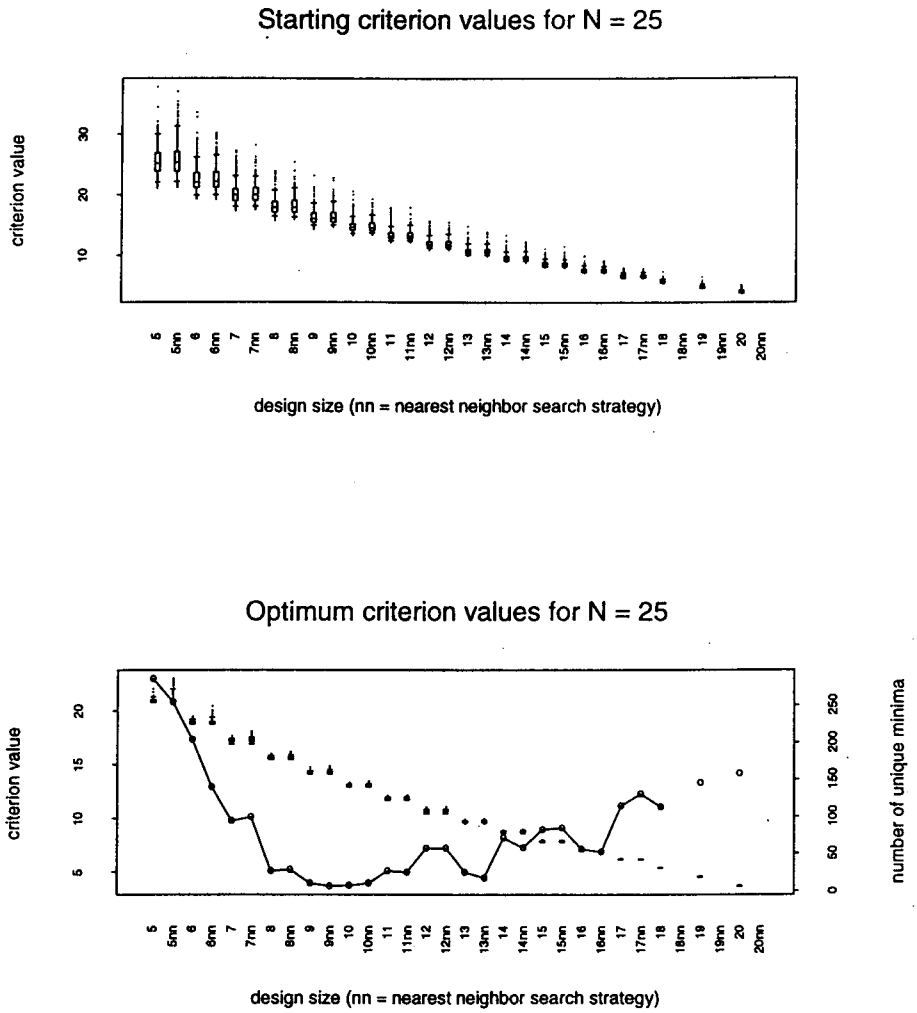


Figure 1: Distribution of starting criteria and optimal criteria for 500 optimizations using a search over all candidates and a nearest-neighbor search on the 25 point candidate set. Right axis on lower plot is the number of optimal designs with the best coverage criterion value. Missing values exist for $p = 18 - 20$ because nearest neighbor search could not be done.

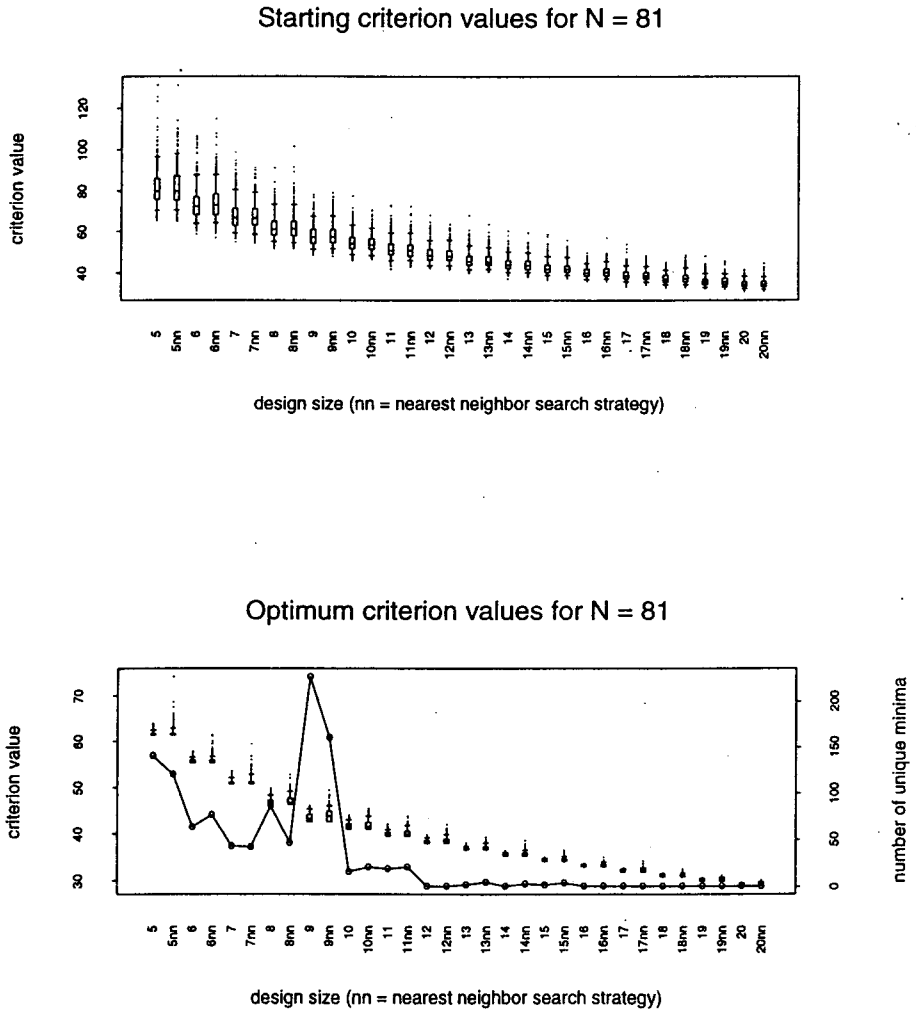


Figure 2: Distribution of starting criteria and optimal criteria for 500 optimizations using a search over all candidates and a nearest-neighbor search on the 81 point candidate set. Right axis on lower plot is the number of optimal designs with the best coverage criterion value.

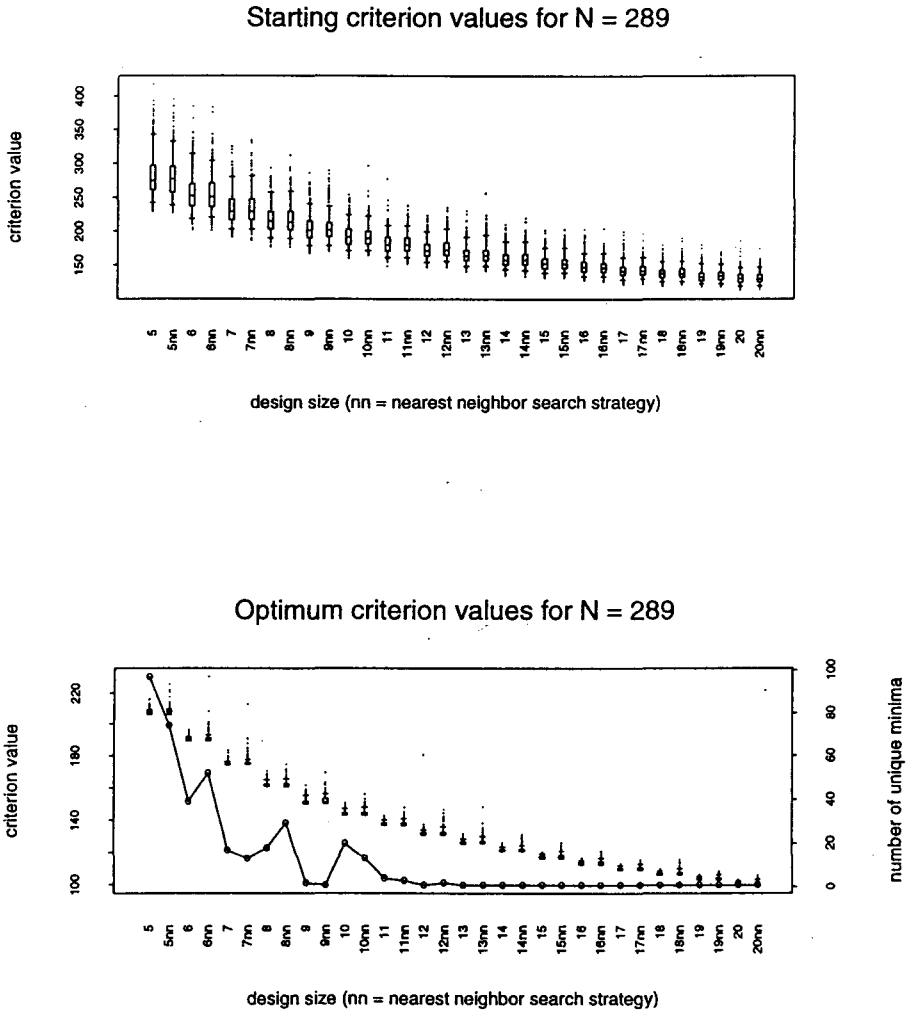


Figure 3: Distribution of starting criteria and optimal criteria for 500 optimizations using a search over all candidates and a nearest-neighbor search on the 289 point candidate set. Right axis on lower plot is the number of optimal designs with the best coverage criterion value.

existing 21 sites.

Another problem is to augment the existing network of 21 sites with 4 additional sites. The best 25 site network in the Chicago region that includes the existing 21 sites is shown in Figure 4(d). One can retain existing points (i.e. prevent them from being considered for swapping out of the design) using the `fixed=` option in `cover.design`. This was done in Figure 4(d). The results for designs computed in Figure 4(c-d) were based on the 720 regular grid points as well as the 21 existing sites, which did not fall on the regular grid of 720 points.

For designs in this example, arc-distance was used to account for curvature of the earth. One may easily modify the distance metric via the `DIST=` option in `cover.design`. For the construction of the Chicago designs, 10 optimizations were conducted and the minimizer of these was used. In all cases, the smallest value of the coverage criterion was found more than once.

6 Conclusions and Discussion

We have presented an algorithm and it's implementation in SPLUS for computing spatial coverage designs. These designs are nonparametric in the sense of being independent of the covariance structure of the process and only depend on the spatial domain being considered and the configuration of the design points. Moreover, computational benefits arise as a result of the criterion being based only on pairwise distances between design locations and candidate locations. The designs are efficiently computed using a simple point swapping algorithm, although efficient recomputation of the criterion for each potential swap is the key aspect of the algorithm. Computing time is reduced by up to 80% for large problems in our study when a nearest neighbor search strategy is implemented. Coverage designs are ideally suited for explicitly discrete problems, such as modification of existing networks and design problems for irregularly shaped spatial domains, because one need only specify a discrete set of points over which to search.

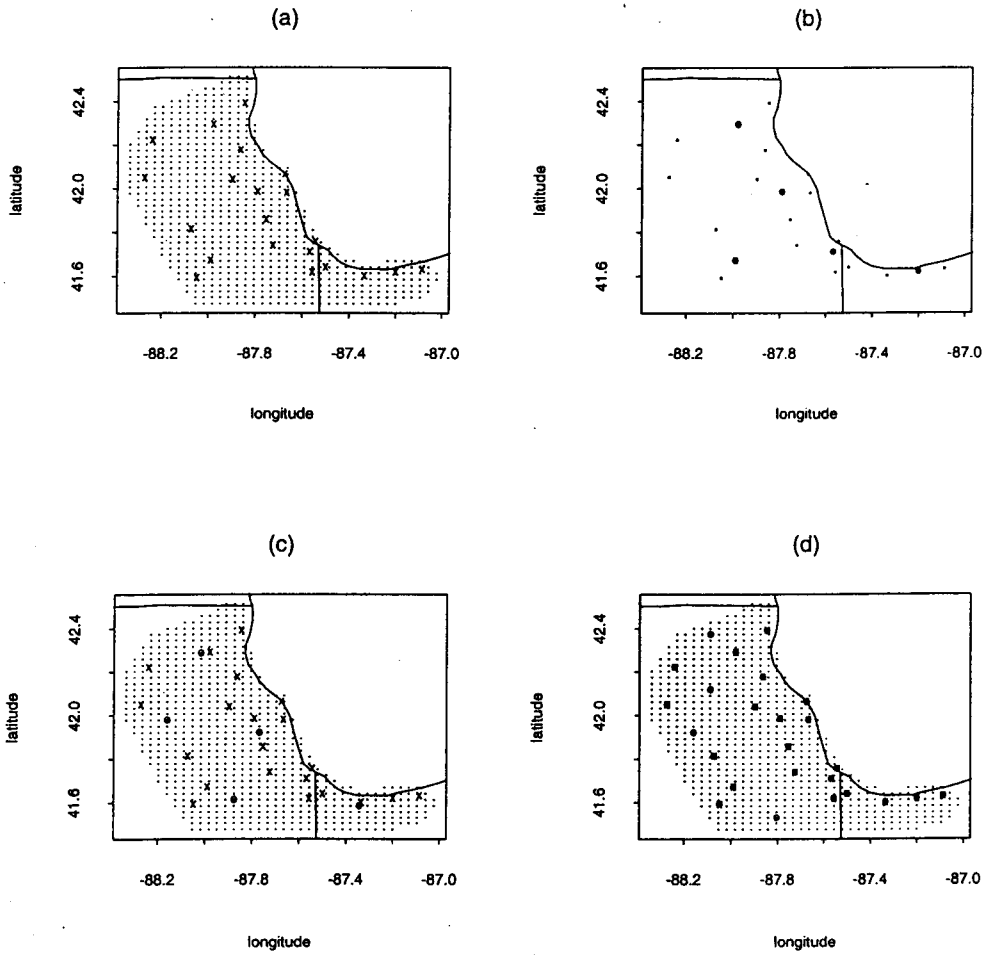


Figure 4: (a) - Location of 21 existing sites and 720 points defining the Chicago region. (b) Existing sites and space filling design of size 5. (c) Best 5 point design over whole region. (d) Best 25 point design that retains 21 existing sites.

Computation of coverage designs using the proposed point-swapping algorithm does not always produce *the* optimal design. Under certain situations, such as a small candidate set and/or a small design size, the optimal design appears to be found often. For large candidate sets and large design sizes, the algorithm doesn't appear to find the optimal design often. However, the best designs found are sufficiently close in terms of the coverage criterion that little difference between the optimal designs and those found via the coverage algorithm would be expected. Similarly, the difference for other criteria (i.e. mean squared error criteria) between these designs should not be substantial. Design exercises for the Chicago region suggest that the swapping algorithm may perform better (in terms of run-time and finding the true minimum) for irregularly shaped regions and for cases with non-uniformly spaced points. No extensive study was conducted to examine this in detail.

The distance metric can easily be changed to allow for modification of the coverage criterion to, for example, designs that account for curvature of the earth and sampling designs in > 2 dimensions. Saltzman et al. (1996) found that coverage designs using a modified distance metric are close approximations to optimal designs. In on-going work we are studying the implementation and effectiveness of the point-swapping algorithm for minimizing covariance-based criteria. Since the coverage designs are good approximations to these "optimal" designs, using a coverage design as the initial starting design in the optimization strategy greatly reduces the amount of computation necessary to compute those optimal designs.

7 Help File and Examples for `cover.design`

This section lists the SPLUS help file and gives some examples for the Splus function, `cover.design` developed to compute coverage designs. `cover.design` is easily added to SPLUS and is available separately or as part of the more comprehensive freeware package `FUNFITS` (Nychka et al., 1996). To get the current version of the design software and the FUNFITS package go to the homepage of Doug Nychka from

the statistics faculty directory at `http://www.stat.ncsu.edu` . The homepage has menu items for this software and other useful links.

The SPLUS help file is included with the `cover.design` code. When placed in the working `.Data/.Help` directory, it can be seen with the `help(cover.design)` command given at the SPLUS command line. The full help file for the version of `cover.design` that goes with this paper is listed here. A more elaborate version of the function, and help file is available with the FUNFITS software.

```
Computes space-filling "coverage" designs.

DESCRIPTION:

        Attempts to find the set of  points  on  a  discrete  grid
        which  minimize  a  geometric  space-filling  criterion.

USAGE:

        cover.design(R, nd, P=-5, Q=1, nruns=1, DIST=e2dist, fixed,
                     start, nn=NA)

REQUIRED ARGUMENTS:

R:      Nxd matrix of candidate points to  be  considered  in  the
        design.

nd:     Number of points to add to the design.  If  points  exist
        and  are  to remain in the design (see "fixed" option), nd
        is the number of points to add.  If no points  are  fixed,
        nd is the design size.

OPTIONAL ARGUMENTS:

P:      A scalar value specifiying a parameter of the criterion to
        be  optimized.  It affects how the distance from a point x
```

to a set of design points D is calculated. P=1 gives average distance. P=-1 gives harmonic mean distance. P=-Inf would give minimum distance (not available as a value). As P gets large and negative points will tend to be more spread out.

Q: A scalar value specifying a parameter of the criterion to be optimized. It affects how distances from all points not in the design to points in the design are averaged. When Q=1 (the default), simple averaging of the distances is employed. Q=Inf (not available as a value) in combination with P=-Inf would give a classical minimax design.

nruns: Number of optimizations to perform. Uses random starts unless "start" is specified.

DIST: A distance metric in the form of an S function. e2dist (see below) computes the 2-d Euclidean norm.

fixed: A vector defining points in R to retain (they are not considered for swapping). Elements of "fixed" correspond to rows of R.

start: An nruns x nd matrix of starting designs. If missing, random starts are used. The number of rows of start MUST equal nruns. Number of columns must be equal to nd. If starting designs are specified, all starting designs must

be specified.

nn: Number of nearest-neighbors to search over. If missing, search is done over all possible candidates.

VALUE:

Returns a list with the following elements:

call: the function call

best.design: (nd+length(fixed)) x d matrix of coordinates in the design.

best.id: rows of R which correspond to points in best.design.

fixed: input data indicating which points were retained.

opt.crit: the optimal coverage criteria value for best.design.

start: matrix of starting design(s) indexed by rows of R.

start.crit: criteria value(s) of starting design(s).

history: describes the swapping history including the points swapped out, points swapped in and the criterion value after each swap.

other.designs: all nruns optimal designs computed. The points in each design are indexed by elements of R.

other.crit: The criterion values for each design in other.designs.

R: candidate points.

DIST: distance metric used.

DETAILS:

Let nd denote the number of design points, dj in the set D, j=1,2,...nd. Let nc denote the number of candidate

points, ci in the set C, i=1,2,...N. The coverage criteria is defined as:

M(D,C) = {SUM_{ci in C}
[sum_{dj in D} ||dj-ci||^P]^(Q/P) }^(1/Q)

Where P < 0, and Q > 0 are parameters. The algorithm used in "cover.design" to find the set of nd points in C that minimize this criterion is an iterative swapping algorithm which will be described briefly. The resulting design is referred to as a "coverage design" from among the class of space-filling designs.

ALGORITHM:

An initial set of nd points is chosen randomly if no starting configuration is provided. The nc x nd distance matrix between the points in C and the points in D is computed, and raised to the power P. Denote this matrix as R. Denote the "row sums" of this matrix as rs_{i} and the vector of row sums as rs. M(D,C) can be computed using these row sums as:

(sum_{i=1,nc} (rs_{i})^(Q/P))^(1/Q) [2]

This representation allows for very efficient recomputation of M(D,C) when the design changes. If point d_{k} is "swapped" for point c_{i}, one must only

recompute the kth column and ith row of the original distance matrix, and NOT the whole matrix.

Thus, for each swap, the row sums vector is updated as

rs.new = rs - R[,k] + ||C - c_{i}||^{p}

The 2nd term is the "old" column of R, and the 3rd term is the "new" column. The ith element of rs.new is replaced by:

rs.new[i] = sum_{j!=k} ||d_{k}-d_{j}||^{p} + ||d_{k}-
c_{i}||^{p}

These elements are the distances between the "new" candidate point, d_{k}, and the "new" design, which includes c_{i}. Finally, M(D,C) is computed for this swap of the kth design point for the ith candidate point using [2]. The point in C that when swapped produces the minimum value of M(D,C) replaces d_{k}. This is done for all nd points in the design, and is iterated until M(D,C) does not change.

When the nearest neighbor option is selected, then the points considered for swapping are limited to the nn nearest neighbors of the current design point.

One can use an arbitrary metric to define the distance between points. This must be in the form of an S

function. For example, the default is e2dist which must be defined by the user as:

```
e2dist<-function(x, y) {
  i <- sort(rep(1:nrow(y), nrow(x)))
  dvec <- sqrt((x[,1] - y[i,1])^2 + (x[,2] - y[i,2])^2)
  matrix(dvec, nrow = nrow(x), ncol = nrow(y), byrow = F) }
```

STABILITY:

The algorithm described above is guaranteed to converge. However, upon convergence, the solution is sensitive to the initial configuration of points. Thus, it is recommended that multiple optimizations be done (i.e. set nruns > 1). Also, the quality of the solution depends on the density of the points on the region. At the same time, for large regions (e.g. > 30 x 30 grids), optimization can be computationally prohibitive using "cover.design" unless the nearest neighbor option is employed.

REFERENCES:

Johnson, M.E., Moore, L.M., and Ylvisaker, D. (1990). Minimax and maximin distance designs. Journal of Statistical Planning and Inference 26, 131-148.

SAS/QC Software. Volume 2: Usage and Reference. Version 6.

First Edition (1995). "Proc Optex". SAS Institute Inc. SAS Campus Drive, Cary, NC 27513.

EXAMPLES: (see below)

7.1 Examples

We assume that the following SPLUS objects exist:

(1) `chicago.locs` : 21×2 matrix of coordinates.

(2) `rdist.earth` : Great circle distance function.

(3) `chicago.region` : 741×2 matrix of coordinates.

The matrix `chicago.region` consisists of the existing 21 points as the first 21 rows, and 720 additional points shown in Figure 4(a) as the remaining rows. The function `rdist.earth` is available from FUNFITS (Nychka et al., 1996). This and all other objects are provided with the SPLUS code for `cover.design`.

First, we consider constructing the design of Figure 4(b), which is the best 5 site network from the existing 21 sites. To create Figure 4(b), type the following commands on the SPLUS command line:

```
> chi.best5 <- cover.design(R=chicago.locs,nd=5,nruns=10,
>              DIST=rdist.earth)
> plot(chicago.locs,pch=''.'')
> library(''maps'')
> map(add=T)
> points(chi.best5$best.design,pch=''o'')
```

To construct the design of Figure 4(c), which is the best 5 site network on the region defined by `chicago.region`, type the following commands on the SPLUS command line:

```
> region.best5 <- cover.design(R=chicago.region,nd=5,nruns=10,
>                 DIST=rdist.earth)
> plot(chicago.region,pch=''.'')
> points(chicago.locs,pch=''x'')
> library(''maps'')
> map(add=T)
> points(region.best5$best.design,pch=''o'')
```

Finally, to construct the design of Figure 4(d), which is the network consisting of the current 21 sites augmented with 4 additional sites, type the following commands on the SPLUS command line:

```
> best25.chi<-cover.design(R = chicago.region, nd = 4, nruns = 10,
>             DIST = rdist.earth, fixed = 1:21)
> plot(chicago.region,pch=''.'')
> points(chicago.locs,pch=''x'')
> library(''maps'')
> map(add=T)
> points(best25.chi$best.design,pch=''o'')
```

For all of these examples, the call to `cover.design` is likely to take some time (see Table 2). One could speed this up by use of the `nn=` option.

Acknowledgements: Portions of this work were funded by Becton Dickinson Research Center, the U.S. EPA under Cooperative Agreement # CR819638-01-0 with the National Institute of Statistical Sciences and the National Center for Atmospheric Research, Geophysical Statistics Project, sponsored by the National Science Foundation under grant #DMS93-12686. The authors thank Perry Haaland at BDRC for supporting the initial work and expanding `cover.design` to it's current state, contained in the software `FUNFITS`.

References

Cox, D.D, Cox, L.H., and Ensor, K.B. (1995). Spatial Sampling and the Environment. Technical Report 18, National Institute of Statistical Sciences, RTP, NC.

Hardin, R.H. and Sloane, N.J.A. (1994). *Operating manual for Gosset: A general purpose program for constructing experimental designs.* AT&T Bell Laboratories, Murray Hill, NJ.

Johnson, M.E., Moore, L.M., and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference* **26**, 131-148.

Kennard, R.W. and Stone, L.A. (1969). Computer aided design of experiments. *Technometrics* **11**, 137-148.

Marengo, E. and Todeschini, R. (1992). A new algorithm for optimal, distance-based experimental designs. *Chemometrics and Intelligent Laboratory Systems* **16**, 37-44.

Mitchell, T.J. (1974). An algorithm for the construction of "D-optimal" experimental designs. *Technometrics* **16**, 203-210.

Nychka, D., Bailey, B., Ellner, S., Haaland, P., and O'Connell, M. (1996). FUNFITS data analysis and statistical tools for estimating functions. Available from Doug Nychka's homepage at `http://www.stat.ncsu.edu` .

Saltzman, N, Nychka, D. and Speckman P. (1996). Connections between the coverage design criterion and prediction variance. Technical Report. National Institute of Statistical Sciences, Research Triangle Park, NC (in preparation).

Nychka, D., Yang, Q. and Royle, J.A. (1997). Constructing spatial designs using regression subset selection. *Statistics for the Environment 3: Pollutation Assessment and Control,* V. Barnett and K.F. Turkman (eds.). Wiley, New York (to

appear).

Tobias, R. (1995). SAS QC Software. Volume 2: Usage and Reference. SAS Institute, Inc., Cary, NC.